Automating the Process of Assigning Students to Cooperative-Learning Teams

Ryan Cavanaugh, Matt Ellis, Richard Layton, Mark Ardis
Rose-Hulman Institute of Technology

Abstract

Assigning students to teams can be a time-consuming process, especially for cooperative learning teams. This paper describes the initial development and testing of a web-based system to assign students to teams using instructor-defined criteria, including criteria consistent with the cooperative learning literature. First, the instructor decides which attributes of students to measure in assigning teams. Next, students complete confidential surveys to determine their attributes. Finally, the instructor assigns a weighting factor to each attribute and the system assigns students to teams. The purpose of the system is to shorten the time to assign teams and to improve the likelihood that teams will satisfy an instructor's criteria for team formation.

The Team-Maker system provides two web interfaces—one for the instructor and one for students. The instructor's interface is used to create the survey and, once students have completed the survey, to assign students to teams in accordance with an instructor-defined weighting scheme. The student's interface allows each student to complete the confidential survey. Features of the team-assignment system important to forming cooperative-learning teams include: the instructor decides which attributes or skills (e.g., grades in prior courses, GPA, writing skill) are to be distributed heterogeneously across teams; the prevention, if possible, of underrepresented minorities being outnumbered on a team; and matching student schedules such that members of a team have a reasonable expectation of being able to meet outside of class.

To test the system, 86 students already assigned to teams by instructors in four sections of a sophomore-level course completed the survey in Spring 2003. In this paper, the teams created manually by the instructors are compared to the teams suggested by the automated system. Results of initial testing are described and plans for future development are outlined. The results show that the system is effective at meeting the instructor's criteria for good team formation and saving the instructor time. The source code for the application is available under an open source license for free distribution and modification.

1. Introduction

1.1 Problem statement

Forming student teams for group work often entails a major time investment for instructors. To make teams according to guidelines given in the cooperative-learning literature, instructors typically design a survey, issue copies to students, collect them, and shuffle the surveys around until satisfactory teams are made. Discussion of the teaching/learning principles underlying this

approach are given in [1]. For a class of thirty students, team creation can take up to an hour, in addition to time spent creating, printing, distributing, and collecting the surveys.

The guidelines for cooperative learning specify how teams can best be made. See, for example, [2, 3, 4]. Student learning is improved by placing students into teams that are heterogeneous in academic achievement, grades in prior courses, writing skills, extracurricular participation, and other criteria. Also, members of underrepresented minorities in engineering should not be outnumbered in a team.

Regarding prior work, a program for assigning teams is described in [5], but it is not as adaptable or user-friendly as the web-based application described here. And no reference was found in the literature to an application created to meet cooperative-learning guidelines.

The goals for the project are:

- Reduce the amount of time needed to create a satisfactory set of teams.
- Increase the likelihood that cooperative learning guidelines (or other instructor preferences for team-formation) are met.
- Provide metrics to assess the extent to which the team-formation criteria are met.

1.2 Solution overview

The primary goal of this project is to save the instructor time by automating the team-assignment process while conforming to the instructor's criteria for team formation. Most of the effort in this project is devoted to developing the team-formation algorithm (described in section 3.1).

The second goal, improving team quality, is difficult to define. A set of heuristics (described in section 3.2) for evaluating the quality of a set of teams is needed. By trying a large number of arrangements of students, the application attempts to find an optimal set of teams. ("Application" denotes the web-based computer application *Team-Maker* that is the subject of this paper.)

The final goal, providing useful metrics on team demographics, is accomplished by generating summaries of student responses on both a per-team and an overall basis. This information is usually not collected as part of the manual team selection process and is a valuable byproduct of the automated system.

The application uses a web-based interface to provide maximum accessibility and cross-platform support for both instructors and students. The application can run on any personal computer capable of serving content on the Internet. One limitation of using the web platform is the inability to provide instant feedback to user input. However, the web platform offers significant advantages in increased accessibility and faster development time.

To use the application, an instructor is given a login by the system administrator. This prevents unauthorized users from creating surveys or viewing confidential data. The instructor then creates a survey by:

1. Specifying the survey respondents (via their email addresses)
2. Creating questions that the students will answer
3. Choosing a deadline

The instructor uses the application to send an email to each student with a customized link (URL) to the survey. Once the students have responded, the instructor specifies how each question is to be treated in forming teams according to the instructor's own criteria, and the application creates a set of teams. Statistical summaries of survey responses are created for each team to assist the instructor in evaluating how well the teams meet the his or her criteria.

2. Instructor and student interaction with the application

2.1 Creating a survey

When an instructor wishes to create a new survey, they log into the Team-Maker system, and are presented with a "home page" that shows all the instructor's previously created surveys. The instructor may then select the survey creation page.

At the survey creation page, the instructor first enters a title for the new survey. This title is used to identify the survey within the Team-Maker software. The instructor then enters a list of email address of all the students that are to take the survey (this list can be edited at a later date, if students add or drop the class). The instructor also sets a deadline for returning the surveys. If a student attempts to fill out the survey after the deadline, they are allowed to proceed, but there is a warning that the due date has passed and that teams may have already been created. The instructor may use a previously-created survey as a template for the new one. If this is chosen, all existing questions from the old survey are copied over to the new one. This allows an instructor to quickly make the same survey for different sections of a class.

Next the instructor adds questions to the survey. To speed up the process the instructor is presented with a box that shows all previously-created survey questions. If none of the existing questions are suitable, the instructor may create a new one. Every question has both a title (used to identify it in the Team-Maker software) and a prompt. For example, a question about a student's age might have the title "Age" and the prompt "How old are you?"

Team-Maker supports four types of questions:

1. Multiple choice: the student picks one item from a list of responses or "I choose not to answer this question".
2. Choose-any-of: the student selects as many items from a list that pertain to him or her (for example, what sports a student plays).
3. Free form response: the student enters text for the instructor to read.
4. Schedule: the student enters times when they are unavailable to work with their team outside of class.

As the survey is being created, the instructor can change the order in which questions are asked, remove a question, or edit an existing question. The instructor may save a survey in an unfinished state and work on it later.

Once the survey is complete, the instructor emails the students asking them to take the survey. The instructor can either notify all participants of the survey or restrict the email to students who have not yet answered the survey. Instructors type a message that goes to each student, and the software adds a customized link allowing each student to take the survey.

2.2 Taking a survey

The email sent out by the instructor contains a link that the student may follow to take the survey. This helps ensure confidentiality of reported data. Students do not need an account for the Team-Maker system: All a student needs to do is click the link contained in the email.

The first question on every survey asks for the student's name to be displayed instead of the student's email address when the instructor is creating teams. Taking the survey is a straight-forward process which mimics taking a survey on paper.

The schedule question creates a grid with checkboxes for hours when the student is not free. Since filling in the grid may be a tedious process, the student may also click an hour and have that hour marked as busy during Monday, Tuesday, Thursday and Friday. (Most classes at Rose-Hulman are on a MTRF schedule.)

After the student has answered all the questions they hit the "Submit Answers" button, and their answers are entered into the system. Students can edit their answers at a later date by simply visiting the survey page again.

2.3 Viewing results

As students respond to the survey, the instructor can view the results of each student, or look at an overall summary of how students responded to each question. When the instructor looks at a student's responses they see the survey exactly as the student did with the answers the student put after each question. The instructor may not edit the survey responses.

When viewing a summary of survey responses, the instructor is presented with a graph for each question showing how the answers spread across the possible responses. Figure 1 shows the graph for a class of 25 students responding to a multiple-choice question in a survey. Four responded they were members of the Phi Gamma Delta fraternity, ten did not answer, and so forth. The bar graph represents the number responses in each category as a percentage of the total number of responses.

Question #10: *Fraternity or Sorority you are a member of (if any)*

(No Answer)	10	48%	
Alpha Tau Omega	1	5%	
Chi Omega	1	5%	
Delta Delta Delta	1	5%	
Phi Gamma Delta	4	19%	
Sigma Nu	1	5%	
Triangle	3	14%	

Figure 1: Summary of responses to multiple choice question

Figure 2 shows the chart reported to the faculty user resulting from a schedule question. It shows, for example, that all of the students are busy 3rd and 4th period on Tuesday and that ten percent are busy on Sunday from 6th through 10th periods. After teams are created, the instructor can view a similar chart for each team, confirming the number of hours outside of class the team members have free in common.

Question #11: *Please check all hours that you are busy this quarter*

0% busy	50% busy	100% busy

Percent busy by hour

	Mon	Tue	Wed	Thu	Fri	Sat	Sun
1	81%	48%	57%	43%	76%	33%	33%
2	67%	71%	24%	71%	67%	33%	33%
3	95%	100%	95%	38%	95%	38%	38%
4	95%	100%	95%	33%	95%	38%	33%
5	33%	95%	29%	43%	33%	29%	29%
6	67%	38%	76%	48%	57%	19%	10%
7	52%	52%	52%	52%	48%	24%	10%
8	57%	71%	67%	57%	48%	24%	10%
9	57%	67%	48%	62%	48%	10%	10%
10	33%	48%	38%	38%	29%	14%	10%
5 PM	38%	33%	43%	33%	29%	19%	14%
6 PM	29%	24%	38%	33%	14%	14%	14%
7 PM	38%	24%	29%	29%	29%	24%	14%
8 PM	38%	19%	14%	29%	19%	19%	10%
9 PM	33%	19%	14%	24%	19%	14%	10%
10 PM	19%	14%	10%	19%	19%	14%	10%

Figure 2: Summary of responses to the schedule question for an entire section.

2.4 Making teams

After responses to the survey have been received from students, the instructor can make teams. The instructor specifies what configuration of team sizes (e.g. 5 teams of 4 and 3 teams of 3; or 7 teams of 3 and 2 teams of 4) to use, which question (if any) should be considered to be the underrepresented-minority question, and how many teams (if any) the instructor would like to specify by hand. Manual team-selection is useful when an instructor knows something that is not provided to the software (for example that two students may not work well together on the same team).

Next, the instructor assigns a weight to each question. For non-diversity questions the instructor chooses a value based on a sliding scale, illustrated in Fig. 3. On one end, the software will attempt to gather students into the same team when they give similar answers to that question. On the other end it will attempt to gather students into the same team when they have different answers to that question. The instructor may also choose to ignore the question for the purposes of team-making. For the diversity question the instructor assigns a weight to each answer that defines how important it is for that particular underrepresented minority to be not outnumbered in a team

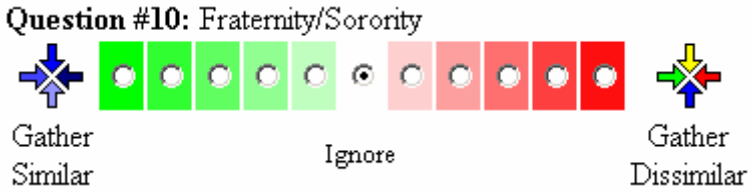

Figure 3: Selecting desired behavior

If the instructor requested to pick some teams manually, a series of drop-down boxes are placed at the bottom of the page. The instructor clicks the "Create Teams" button and the team-making algorithm creates teams. After teams are created, the instructor can either save the teams or go back and tweak the weights assigned to each question. Once the initial set of teams has been created, the instructor may swap people between teams manually.

Instructors are free to implement any team-formation criteria they choose. The application has no built-in assumptions about what attributes to include in the survey nor what weight to assign to any particular attribute. These decisions are up to the instructor. One may choose to have students assigned to teams based on similar or dissimilar responses to any question.

3. Team-making algorithm

3.1 Team creation

The team-making algorithm is based on a concept of a weighting for each question. Weight is specified by a number that indicates how the question should be treated (the sign of the number), and how important it is (the magnitude of the number). A weight of zero indicates the question should be disregarded. A negative weight indicates that students with *different* responses should

be placed in the same team, where a positive weight indicates that students with *similar* responses should be placed in the same team.

At a high level, the goal of the algorithm is to generate the best overall score possible. The overall score is equal to the lowest score of all of the teams. The score for each individual team is the sum of the weight of each question multiplied by the score for that question. The score for the question is determined by the heuristic for the type of question. The heuristic function for each type of question is defined in the next section.

To accomplish this goal, the algorithm does the following:

1. Allocate the specified number of teams
2. Place students randomly into a non-full team until all are placed
3. Evaluate the heuristics and compute an overall score for the section
    a. Exchange (swap) students between two teams
    b. Compute a new overall score. If the score is higher, keep the swap; if lower, undo the swap.
    c. Iterate steps (a) and (b) until no swaps are successful
4. If the new set of teams is better than the old set, replace the old set with the new set
5. Repeat steps 1 through 4 a fixed number of times (usually 50)

This is a routine implementation of the hill-climbing algorithm [6], which efficiently finds local maxima but is generally poor at finding global maxima. By repeating the algorithm many times, the global maximum is more likely to be found. In order for the hill-climbing strategy to effectively find a global maximum, it must start in different places in the search space. The random allocation of students in step 2 provides for a random starting location in the search space. Repeating the outer loop 50 times was found to be a good number for nearly always finding an optimal set of teams while keeping execution time low.

3.2 Heuristics

For each question type, there is a different heuristic used in determining the score. Every heuristic returns a value in the range [0, 1], with 1 representing a homogeneous composition and 0 representing a heterogeneous composition. It is important to note that the goal of the question-level heuristics is *not* to distinguish "better" from "worse," but to determine "same" from "different."

The success of the algorithm in general is almost entirely dependent on the definitions of the heuristics. An effective heuristic for use in this algorithm has several properties: continuity, linearity, and range.

A perfectly continuous heuristic is capable of producing as many different values as there are distinguishable inputs. For example, a team of four students answering a yes/no question represents 4 different states, so the heuristic should be capable of producing 4 different values. If input states are equivalent under the cooperative learning guidelines (for example, 1 answering yes and 3 answering yes might be the same depending on the question) then the outputs should be the same.

A linear heuristic accurately represents small changes to the input as small changes to the output. For example, a heuristic that returned 0.1, 0.2, 0.3, or 1.0 would probably be more effective if 0, 0.33, 0.66, and 1.0 were returned instead. This is not always entirely practical since we consider many things to be non-linear in definition. For example, in a team of four when trying to minimize the number of team members belonging to the same fraternity, we consider the change from 2 to 3 members of the same fraternity in the team to be a larger change than from 1 to 2.

Heuristics that use the full range of the output values will be more effective than those that only use a smaller interval of the full range. Many of the heuristics defined below would have a minimum value of the inverse of the number of students in the team, that is, the minimum possible value is not zero. This effectively means that there is no "perfect" set of teams. Instead, the case that returns the lowest value is set to return zero. This gives a full range to all heuristics, resulting in equal importance being given to each question until the instructor-assigned weights are imposed.

3.2.1 Overall heuristic

The overall heuristic returns the score of the worst-scoring team. Since only two teams are altered in each step, the algorithm can make an efficient optimization by comparing to see if the minimum of those two teams improved in determining whether or not to keep the swap, which is equivalent to evaluating the entire overall heuristic and comparing to the old value.

$$
\begin{aligned}
F(\bar{t}) &= \min\{T(t_1), T(t_2), \ldots, T(t_n)\} \\
\bar{t} &= \text{The set of all teams} \\
T(n) &= \text{The score for team } n \\
t_n &= n^{\text{th}} \text{ team}
\end{aligned}
$$

3.2.2 Multiple choice

The multiple choice heuristic scores a team containing multiple people answering the same response to the question in a quadratic manner. The result is normalized by dividing by the number of people in the team.

$$
\begin{aligned}
H(t,q) &= \frac{\sum_{i=1}^{|R|} \bigvee_{j=1}^{|t|} A_{i,j}}{|t|} \\
t &= \text{Set of all students in the team} \\
A_{r,j} &= \begin{cases} 1 & \text{student } j \text{ responded yes to option } r \\ 0 & \text{otherwise} \end{cases}
\end{aligned}
$$

This heuristic determines the heterogeneity of a given team by counting the total number of groups represented in that team. For example, if a team is composed of two Minnesotans, two Californians, and one New Yorker, the total number of states represented is three out of a possible five, so the score is 3/5 = 0.6.

3.2.3 Choose-any-of

The choose-any-of heuristic scores teams according to the sum of squares of students answering "yes" to each possible response. The heuristic determines homogeneity by making several assumptions. For example, if the question asked is which sports a student involved in, it is assumed that having one football player is the same as having no football players. Two football players on a team is weighted less heavily than three or four football players on the same team. For each possible sport in the question, this process is repeated and the results are summed.

$$
\begin{aligned}
H(t,q) &= \max\left\{1 - \sum_{i=1}^{|R|} \frac{[d(G(R_{q,i},t))]^2}{|t||R|}, 0\right\} \\
G(r,t) &= \sum_{j=1}^{|t|} A_{r,j} \\
d(r) &= \begin{cases} 0 & x \le 1 \\ x & \text{otherwise} \end{cases} \\
t &= \text{Set of all students in the team} \\
R_{q,i} &= \text{The } i^{\text{th}} \text{ response to question } q \\
A_{r,j} &= \begin{cases} 1 & \text{student } j \text{ responded yes to option } r \\ 0 & \text{otherwise} \end{cases}
\end{aligned}
$$

3.2.4 Schedule

The schedule heuristic counts the total number of hours in which all students in the team are available. The arbitrary value $k = 40$ represents the number of hours after which additional times when students are all free no longer counts as being more homogeneous.

$$
\begin{aligned}
H(t,q) &= \max\left\{\frac{\sum_{i=1}^{|P|}(1 - \bigvee_{j=1}^{|t|} b_{i,j})}{k}, 1\right\} \\
t &= \text{Set of all students in the team} \\
P &= \text{Set of hours in the week} \\
b_{i,j} &= \begin{cases} 1 & \text{student } j \text{ is busy at time } i \\ 0 & \text{otherwise} \end{cases} \\
k &= 40
\end{aligned}
$$

3.2.5 Underrepresented minority

The minority heuristic is a special case because it is allowed to exceed the [0, 1] bound. The input weights to the minority heuristic range from 0 to 5, as does the output value. When a member of a minority is present as the sole representative of that minority in his or her team, the

negation of the weight is returned. If there are two or more members of that minority in the team, the positive value of the weight is returned.

$$
\begin{aligned}
M(t,q,u) &= \sum_{i=1}^{|R|} \Big(u_i \cdot N(R_{q,i}, t) \Big) \\
N(r,t) &= P\Big(\sum_{j=1}^{|t|} A_{r,j} \Big) \\
P(x) &= \begin{cases} 1 & x \geq 2 \\ -1 & x = 1 \\ 0 & x = 0 \end{cases} \\
t &= \text{Set of all students in the team} \\
R_{q,i} &= \text{The } i^{\text{th}} \text{ response to question } q \\
u &= \text{The set of weights for the minority question} \\
u_i &= \text{Weight assigned to minority response } i \\
A_{r,j} &= \begin{cases} 1 & \text{student } j \text{ responded yes to option } r \\ 0 & \text{otherwise} \end{cases}
\end{aligned}
$$

4. Automated team-assignment results

To compare the effectiveness of the computer algorithm against a manual team assignment, data from four sections of a sophomore-level engineering course was entered, and teams were generated. The class used was a sophomore-level dynamics course, ES205 Analysis and Design of Engineering Systems. This course is the last of five required engineering science courses for mechanical and electrical engineering students in the sophomore engineering curriculum at Rose-Hulman Institute. This five-credit-hour course has four hours of lecture and one three-hour lab per week. Students are assigned to teams for lab work both in side and outside of the lab period. Student demographics for these four sections are shown in Table 1. Here "minorities" refers to African-American, Hispanic, Asian-American and other underrepresented racial/ethnic minorities in engineering.

Table 1: Demographics of the sample course

N	No. of teams	Men	Women	Non-minorities	Minorities
86	24	87%	14%	94%	6%

During the quarter the course was taught (Spring 03), teams for all sections were assigned manually by the instructors in the normal course of teaching the class. These "manual" team assignments are compared below to the team assignments created by the Team-Maker program. The automated assignments were made for the comparison made in this paper and were not used in the course.

The resulting heuristic scores are compared below in Tables 2 through 4. Higher scores are better; the score is unitless and is calculated by the heuristics in section 3.2.

Table 2 shows that the mean heuristic score for automated team creation is greater than the mean score for teams created manually by 7.5%. Thus, to the extent that the heuristic scores really do represent how well teams are made, the automated system creates a better set of teams than the instructors did.

Table 2: Comparing automated to manual team assignments, average for all sections

Team formation	Mean	Min	Max	Std Dev
Automated	32.43	29.83	34.50	1.27
Manual	30.01	23.08	39.42	4.31

This result shows that according to the heuristics defined above, the mean score for the teams assigned by the program is higher than that of those assigned manually in this instance. Given the nature of the algorithm, this is an expected result, since it is designed to find maxima. The closeness of the automated score to the manual score shows that the heuristics are a good approximation of what the educator in this case considers when making teams.

Assuming the scores are normally distributed, the mean automated score is 32.43±0.55 and the mean manual score is 30.01±1.86 , both with a 95% confidence level. This gives a range of [31.88, 32.98] for the automated teams and [28.15, 31.87] for the manual teams. Since the confidence intervals do not overlap, the difference of 2.42 in the means is (just barely) statistically significant. In addition, the standard deviation for the manual teams is much higher than that of the automated teams. Thus, the application creates teams that meet the instructor-assigned criteria more uniformly than those manually assigned by the instructor.

Table 3 shows the results for each section. For every section, the automated heuristic score is greater and the standard deviations are smaller than the scores and deviations for the teams created manually.

Table 3: Comparing automated to manual team assignments, by sections

Section	Automated score (std. dev.)	Manual score (std. dev.)
5	33.07 (0.93)	31.64 (5.11)
1	32.65 (1.46)	29.03 (3.77)
6	32.38 (0.56)	29.32 (4.23)
2	30.88 (0.90)	29.30 (4.17)

Table 4 shows a similar comparison for each team, sorted in order of decreasing automated scores. In 19 of 24 cases, the automated team-formation has a higher heuristic score, indicating that the automated system meets the team-assignment criteria just a bit better than the instructor does when assigning teams manually. In five cases (shown shaded), the instructor created

individual teams with higher scores than the automated teams. However, as already shown in Table 3, the instructor never had a section mean greater than the mean of the automated teams.

Table 4: Comparing automated to manual team assign7ments, by team.

Section	Team no.	Automated Score	Manual Score
1	6	34.50	34.08
5	8	34.25	39.42
5	6	33.75	34.17
5	7	33.75	36.75
1	5	33.75	32.25
5	5	33.58	32.25
6	5	33.42	33.67
1	4	33.33	29.65
6	4	33.08	31.67
5	4	32.83	30.42
5	2	32.50	26.08
5	3	32.50	29.67
6	3	32.50	31.08
2	5	32.25	36.67
1	3	32.00	28.17
6	2	32.42	27.08
6	1	32.00	23.08
1	2	31.74	25.92
5	1	31.42	24.33
2	4	31.17	28.42
2	3	30.67	27.67
1	1	30.58	24.08
2	2	30.50	27.00
2	1	29.83	26.72

5. Configuration and distribution

The source code for the project is available under an open source license for free distribution and modification. Platform requirements for setting up a server, source code and configuration files, developer documentation, and other information can be found at http://teammaker.cs.rose-hulman.edu.

The application is designed to work at any institute, so parameters such as the layout of the schedule grid, the start of the calendar week, administrator email addresses, and others are configurable by whoever sets up the system.

Educators wanting to evaluate the system on a trial basis may obtain an account by emailing richard.layton@rose-hulman.edu.

6. Conclusions

Regarding the primary goal of saving the instructor time, the system is very effective. It is estimated that creating the first survey takes around 30 minutes, which is comparable to the time required to create a paper survey. Issuing the survey to students, collecting the surveys, and creating teams take at most ten minutes each. Incremental costs of adding more students are low

in the Team Maker system, and high in the manual method. Overall, the automated method saves about an hour of instructor time for each section of thirty students.

The application is also effective at creating teams that meet the instructor-defined team-formation criteria. The results show that for each section the mean automated score is slightly better than the mean manual score and that the increase in the mean score is significant. More importantly, the software does no harm, that is, the automated team selection does no worse than the manual selection, and in less time.

The reports and summaries such as Figures 1 and 2 are valuable and informative assets made available to the instructor. These would be impractical to generate from a paper survey, and are a worthwhile byproduct of the Team-Maker system.

7. Future work

More work is necessary in determining exactly how each of the heuristics should be defined. A better understanding of how instructors value the properties examined by the heuristics will lead to a more consistent selection of teams with a high probability of successful learning experiences. Also, the algorithm used to form the teams would be an excellent practical application of advanced artificial intelligence techniques.

Future work includes additional testing of the application with faculty and students at Rose-Hulman and at other institutions. Once acceptance testing on the application is complete, a study can be performed to determine if the teams created by the automated system perform better than teams created manually.

Acknowledgements

Our thanks to the ASEE ERM Minigrant program for funding. The initial version of the Team Maker application was written as part of CS414 and CS415 Software Engineering I and II in the 2002-2003 school year. The students on the project were Ryan Cavanaugh, David Aramant, Mark Newheiser, Brian Klimaszewski, Brian Kopecky, and Robert Drake, advised by Don Bagert. Students in ES205 and ME311 participated in testing the system and provided valuable test data. Funding for a new server was provided by Rose-Hulman Institute.

Bibliography

1. Felder, R.M., Stice, J.E. and Brent, R., 1999, Workshop Notebook*, National Effective Teaching Institute* (NETI), Charlotte, NC.
2. Smith, K.A., 2000, *Project Management and Teamwork*, McGraw-Hill, Boston, MA, p. 15-17.
3. Felder, R.M., 1993, Reaching the second tier—learning and teaching styles in college science education, *J. College Science Teaching*, **23**(5), pp.286-290
4. Johnson, D.W., Johnson, R.T., and Smith, K.A., 1991, *Active Learning: Cooperation in the College Classroom,* Interaction Book Co., Edina, MN.
5. Redmond, M.A. (2001) A computer program to aid assignment of student project groups, *SIGSCE Bulletin*, Assoc. for Computing Machinery (ACM), pp. 134-138.
6. Russell, S. and Norvig, P, 1995, *Artificial Intelligence A Modern Approach*, Prentice-Hall.

RYAN CAVANAUGH
Ryan Cavanaugh, formerly a Computer Science student at Rose-Hulman, graduated in February, 2004. He is now working at Microsoft's Visual Basic team after three internships at the company. His interests include compilers, ray-tracing, and game simulation.

MATT G. ELLIS
Matt Ellis is a junior Computer Science student at Rose-Hulman, graduating in May of 2005. He plans to pursue a Ph.D. in Computer Science upon graduating. His interests include programming languages, compilers, and artificial intelligence.

RICHARD A. LAYTON
Richard Layton received his Ph.D. from the University of Washington in 1995 and is currently an Assistant Professor of Mechanical Engineering at Rose-Hulman. Prior to his academic career, Dr. Layton worked for twelve years in consulting engineering, culminating as a group head and a project manager. He is a member of the Teaching Workshop Group of the ERM Division of ASEE, giving workshops on building student teams.

MARK A. ARDIS
Mark Ardis received his Ph.D. from the University of Maryland in 1980 and is currently a Professor of Computer Science and Software Engineering at Rose-Hulman. Dr. Ardis has also taught at the University of Illinois, Wang Institute, and Carnegie Mellon University. Prior to joining Rose-Hulman he spent nine years performing research on software engineering methods at Bell Laboratories.